IBRAC: Image-Based Rendering Acceleration and Compression

Ilmi Yoon and Ulrich Neumann
Computer Science Department
Integrated Media Systems Center
University of Southern California
Los Angeles CA 90089

ABSTRACT

This paper presents an image-based rendering technique that has several benefits to remote or web-based rendering systems. It adapts an image-based rendering acceleration method to the direct rendering of compressed images. The approach is attractive for remote rendering applications where a client system may be a relatively low-performance machine and limited network bandwidth makes transmission of large 3D data impractical. The efficiency of the server and client generally increases with scene complexity or data size since the rendering time is predominantly a function of image size.

This technique uses stored images to accelerate the rendering and compression of new synthetic scene images. The algorithm employs ray tracing and epipolar constraints to exploit spatiotemporal coherence between the current and previously rendered images. The reprojection of color and visibility data accelerates the computation of new images. The rendering method intrinsically computes a residual image, based on a user specified error tolerance that balances image quality against computation time and bandwidth. Encoding and decoding uses the same algorithm, so the transmitted residual image consists only of significant data without addresses or offsets. We measure rendering speed-ups of four to seven without visible degradation. Compression ratios per frame are a factor of two to over ten better than MPEG2 achieves in our test cases. There is no transmission of 3D scene data to delay the first image.

Keywords : Image-based rendering, compression, MPEG, remote-rendering, ray-tracing, internet application

1. INTRODUCTION

Computer graphics algorithms synthesize images from 3D models or 2D images. Despite advances in the hardware and algorithms for 3D-model rendering, the diversity of graphics applications reveals algorithmic deficiencies and identifies new problems. For example, graphics for Internet applications spurred the development of VRML for remote interaction with, and compression of, 3D models [Djurcilov98] [Earnshaw97] [VRML]. Complex 3D models are commonplace, and despite the progress in VRML revisions and geometry compression [Deering95] [Li98], the compression and transmission of large models remains a challenging problem. Model transmission introduces startup up latency, and after transmission, the rendering performance is unpredictable since it depends on the client configuration. The transmission of images rather than models reduces startup latency, however, the bandwidth requirements for images depend on the compression method. Compression methods are mainly designed for video, not graphics. Video images differ from most graphics in their spectral and noise content, often masking artifacts that are clearly visible in graphics images. Furthermore, motion-based compression methods (e.g., MPEG) that take advantage of spatial and temporal (spatiotemporal) coherence require one or more sets of complete rendered images before compression can begin. An important motivation behind the IBRAC method is to increase rendering and compression efficiency by directly rendering compressed images.

Image-based rendering (IBR) introduces motion prediction within the rendering process. IBR trades transformations and shading computations for re-projection or warping of images [Chen93] [McMillan95]. The efficiency of IBR generally increases with scene complexity, since the rendering time is predominantly a function of image size. As an example, in view interpolation [Chen93] the motion of pixels (i.e. the optical flow) from one camera position to another produces smooth interpolated images. Neighbor-pixel interpolation fills the holes that arise in new images from occlusion or subsampling. Refinements of the hole filling approach use additional scene knowledge, such as layered depth images [Shade98]. A weakness of many IBR methods is the lack of freedom to manipulate individual objects, which stems from the lack of object information in the image representations.

IBR approaches fall in two classes. The first class produces novel images from photographic data [McMillan95] [Chen95] [Laveau94] [Levoy96]. In this class, no 3D models are necessary and the mappings from scene-data images to new images are 2D to 2D through implicit 3D relationships. The second class uses a combination of 2D images and 3D models to produce novel images [Torborg96] [Debevec96] [Lengyel97] [Regan94]. These approaches focus on increasing the rendering efficiency or realism by using images, and we refer to them collectively as *hybrid IBR* techniques. Our method extends the hybrid class, using images to simultaneously accelerate interactive rendering and produce a compressed representation for archival storage or transmission.

Forerunners of hybrid IBR techniques include texture maps, bump maps, and reflection maps or environment maps. Texture maps approximate geometric complexity with color variations [Blinn76] [Greene94]. Bump maps model surface roughness or wrinkling by surface-normal variations [Blinn78] [Cook84]. Reflection or environment maps approximate global illumination effects with a hemispheric image projection. Regan and Pose [Regan94] developed a hybrid system that reuses portions of a previous image or a reference image by address recalculation, thereby effecting a simple reprojection of an image texture and increasing the rendering efficiency. The Talisman system and coherent layer rendering extend these ideas [Shade96] [Tor-borg96] [Lengyel97]. They exploit temporal coherence for individual objects by using sprites or image-layers. Affine warps model an object's image variations, and fidelity metrics determine when a sprite requires re-rendering from a 3D model. In [Mark 97], post-rendering 3D warping exploits similar ideas to achieve low latency from the acceleration. In [Yagel93], temporal coherence is exploited in volume rendering. Each pixel keeps the first opaque voxel position and forward projects it to leap over transparent voxels. These hybrid IBR methods depend on both spatial and temporal coherence to efficiently produce consecutive images of smooth synthetic-scene animations.

Our work extends hybrid IBR techniques to compression. We reason that since IBR methods extract spatiotemporal coherence between images for acceleration, the same coherence data can also serve as motion prediction for compression. Image compression methods search for coherence in 2D image sequences, after the rendering (or imaging) process. IBRAC merges the rendering and compression functions to accelerate rendering of compressed images.

This paper focuses on compression and acceleration by employing a backward projection to exploit temporal coherence. Generally, spatial coherence (a form of redundancy) is encoded by the Discrete Cosine Transform (DCT) used in JPEG compression [Rabbani91] [Wallace91]. Temporal coherence arises from the small motions that occur between smoothly animated frames, and (block-based) motion prediction encodes this coherence in MPEG compression [Hidaka90] [Liou91] [MPEG]. Techniques in JPEG and MPEG are targeted at general images, IBRAC takes advantage of significant information available during the rending synthetic images.

Guenter [Guenter93] and Yun [Yun97] used transformation information available in animation scripts and geometry data in each pixel to enhance the compression ratio of sequences of pre-rendered images. Pixel motion is predicted by the script transformations, and the difference between the predicted and target images is stored in a residual image. The residual image is sparse and has low entropy, so compression of the residual with entropy coding methods achieves a factor of 1.4 over JPEG performance.

In a compression strategy targeted at remote rendering, a high-performance graphics engine acts as a server, and a low-end workstation as a remote client [Levoy95]. The client renders low-quality images while the server renders both low- and high-quality images, transmitting only compressed residual images to the client. This approach assumes that the client first receives a copy (or simplified version) of the scene data before rendering begins. The server does not benefit from any acceleration; in fact, the server computes a high quality image, a low quality image, a residual image, and a compressed residual image.

In [Cohen-Or 97] [Mann 97], another strategy is presented for the partition of the rendering task in which the client is able to generate several frames autonomously without the transmission of residual images. The client receives visible portions of 3D data (only geometry, not textures) and an initial image. Images are rendered into a Z-buffer and backward-projected to the initial image to fetch the texture color. As a tradeoff between bandwidth and image quality, only selected pixels are transmitted to perform corrections in the most significant areas. To predict the area, relevant portions of model data are identified, transmitted and rendered in the client.

IBRAC computes pixel motions by searching prior images (with depth information), but unlike MPEG, which computes motions from consecutive pre-rendered images, our system computes motions from any prior image and a related transformation. The IBRAC method is applicable to both predetermined sequences and interactive control (e.g., remote visualization or

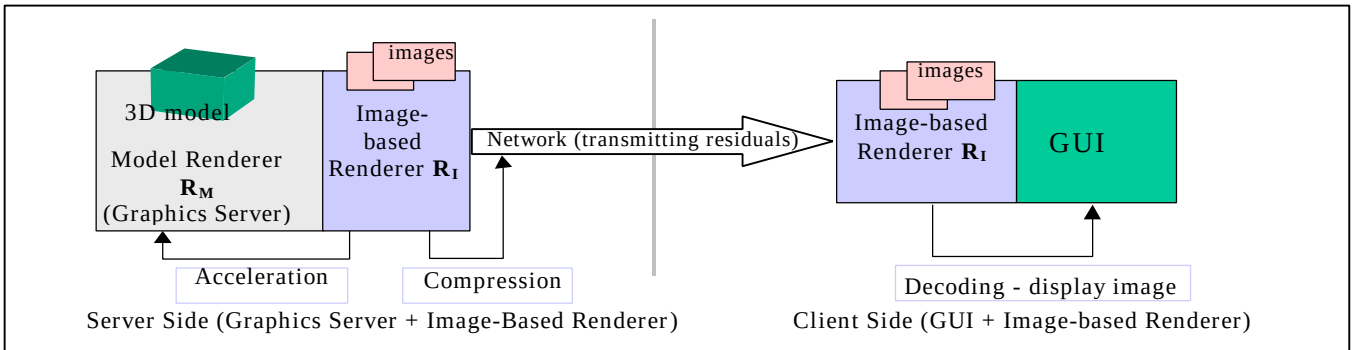


Fig 1. – IBRAC Remote Rendering System. The image-based rendering system on the server side accelerates graphics rendering of compressed images. The image-based rendering system on the client side decodes the compressed data stream for display.

Internet applications). In an IBRAC remote rendering application, the server has the 3D model and the remote client does not. The fact that no model needs to be resident or transmitted to the client is significant since it reduces the startup bandwidth and delay in displaying the first remote image. The performance of an IBRAC system with a given image size is predominantly a function of the server's accelerated rendering time, not a function of client's rendering capability or model data size.

The following system features are desirable for interactive remote visualization and Internet applications:

- The 3D model should reside at the host and not be transferred to the remote client.
- Minimal network bandwidth should be consumed for each frame, whether interactively defined or replayed from a predefined animation.
- Modest computing resources are expected at the remote client.

For archival storage applications, minimal bandwidth is the essential attribute. The bit stream is simply stored rather than transmitted.

Our IBRAC system uses a hybrid IBR method that accelerates image synthesis by a factor of 4-7 in test cases [Yoon97]. The server and client run synchronized versions of the same renderer, predicting pixel motions from prior images (Fig. 1). The server computes residual pixel values for low confidence predictions and transmits them to the client, yielding compression ratios 4-10 times higher than MPEG1 achieves for the same pre-computed images.

The remainder of this paper is organized as follows: Section two describes the algorithm for accelerated rendering. Section three presents the integration of the rendering and compression system. Finally, we present results and discuss future research directions in section four.

2. RENDERING ACCELERATION

2.1. Model and Image-Based Rendering Subsystems

The hybrid IBR system includes two rendering subsystems. One is based on ray casting as described by Appel [Appel67] and enhanced by Whitted [Whitted80]. Rays cast from a camera position pass through pixels of the image plane, intersecting 3D objects in the scene. This process, called the model renderer ($\mathbf{R_M}$), has access to the 3D geometry as shown in Fig. 2.

The second element is an image-based renderer ($\mathbf{R_I}$) that tests for ray intersections in prior rendered images (called reference images) with depth values at each pixel (Fig. 2). Ray intersection tests in pixel-based scene representations are prone to uncertainty, so heuristics classify the results into three responses: *definite miss, definite hit, or possible hit*. A definite miss indicates that the ray missed all objects in the image. A definite hit returns the estimated 3D position of the intersection as well as a color estimate at that point. A possible hit returns the estimated 3D position of the intersection, allowing the model renderer to proceed from that point. High spatiotemporal coherence results in relatively few possible-hit

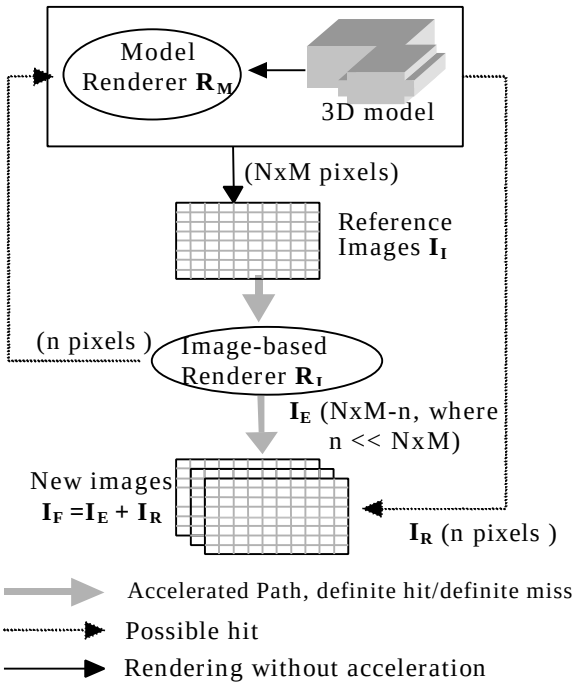


Fig 2. – Rendering Acceleration : Among NxM pixels of a new image, the majority of them (NxM-n) can be derived from previously rendered images using the image-based renderer, and the remaining pixels (n) are rendered by the model renderer. Since the IBR speed is higher than the speed of the model renderer, the overall rendering speed is accelerated.

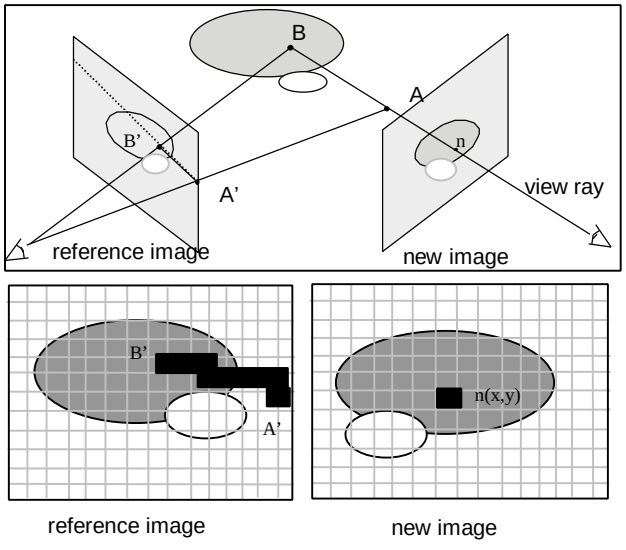


Fig. 3 – The image-based renderer casts a view ray from a pixel n, and the ray is projected to a line (A'B') in a reference image. The reference image line is traversed and tested for intersections with the sampled geometry.

occurrences, so the model renderer is rarely invoked. For complex scenes, the ray intersection search performed by the image-based renderer is considerably faster than the ray-intersection tests performed by the model renderer, however pixel-based intersection tests are not robust, so the model renderer is needed to ensure fidelity. The two renderers cooperate as a hybrid IBR system to efficiently compute new images, as shown in Figure 2 and detailed in the remainder of this section

2.2. IBR Intersection and Classification

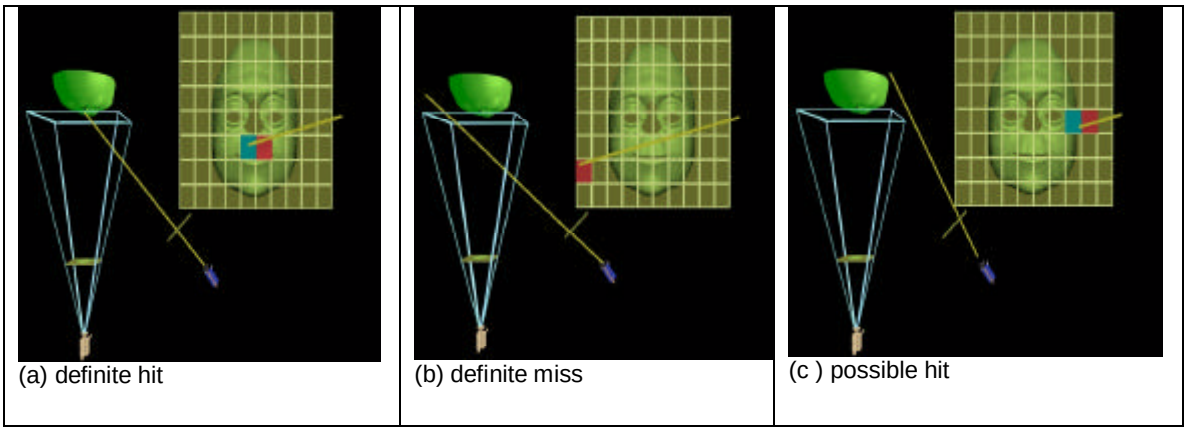
The image-based renderer back-projects new-image rays into previously rendered (reference) images [Laveau94]. For each pixel, a view ray AB is projected on to the reference image A'B' and searched for intersections (Fig.3). A view ray AB can be expressed as a vector originated from a new camera position Cn to the objects through a pixel n(x,y) :

direction(AB) = $Pn^{-1} \bullet n(x,y) - Cn$,
Pn is the projection matrix of the new image,
$n(x,y) = Pn \bullet A$, and $n(x,y) = Pn \bullet B$,
and Pr is a projection matrix of the reference image.
$B' = Pr \bullet B$, and $A' = Pr \bullet A$
direction(A'B') = direction($Pr \bullet B - Pr \bullet A$)
= direction($Pr \bullet Pn^{-1} \bullet n(x,y) - Pr \bullet Cn$)
= $(\nabla x, \nabla y, \nabla z)$

Define depth values of the reference image pixels as $depth_R(index)$. Surfaces may be found by traversing the reference image starting from $(Pr \bullet Cn)$ along the direction of A'B' by $(\nabla x, \nabla y, \nabla z)$. If there is no object between the new camera position and the reference image view frustum, start the ray within the reference image (A' in figure 3). Intersection tests compare the ray depth at every step $(z(A') + \nabla z/(\nabla x \text{ or } \nabla y))$ with depth values at the reference image pixels ($depth_R(index)$) traversed by the ray. A pixel depth value is a point sample in the center of the pixel. A ray depth is sampled at the closest point to the pixel sample position. All depth values are in the same perspective space. If the pixel depth value ($depth_R(index)$) is less than the ray depth $(z(A') + \nabla z/(\nabla x \text{ or } \nabla y))$, then we assert that the ray has passed behind a surface, causing an intersection. The intersection position determines the motion of the pixel between the new image and the reference image.

```
Intersection Test (viewray AB)
    projected_viewray A'B' = projectToImage(AB);
    Pixel p = first(A'B');
    Index i = index(p);
    While (p.depth <  depth_R(i) ) do
        prev_p = p;
        p = next(A'B');       // step 1 pixel in x or y direction and add ∇z/(∇x or ∇y)
        i = index(p);
    if(prev_p.object == p.object && prev_p.depth - p.depth < threshold)
        return (definite hit, p);
    else if (outside_image(p) && p.depth > max_depth_of_scene)
        return (definite miss, p);
    else return (possible hit, p);
```

The pixel motion calculation also facilitates an estimation of motion confidence. A confidence metric is essential for determining if residual-image pixels must be computed. Confidence values correspond to the intersection cases illustrated in figure 4 as *definite miss, definite hit or possible hit.* A definite hit is indicated (Fig. 4a) when a ray transitions from in-front-of to behind a surface. A definite miss (Fig. 4b) occurs for a ray that fails to pass behind any pixels. A possible hit occurs when a ray suddenly appears behind a surface it was never in front of (Fig. 4c), or when a ray exits the image before leaving the scene's bounding volume. For reasons explained later in this section, a possible hit also occurs when the depths of the two neighboring pixels in the reference image differ by more than a threshold. Definite misses and definite hits are high-confidence motion estimations. Misses return a background color and depth. Hits return the pixel color and depth, interpolated from neighboring samples. Possible hits are low-confidence motion estimations and their positions are provided to the

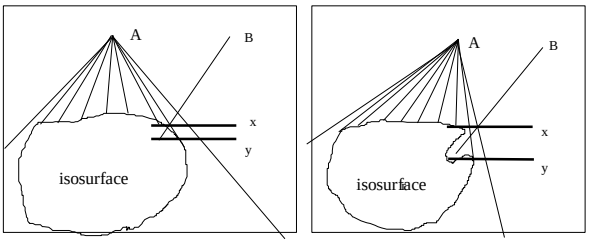


(a) definite hit (b) definite miss (c) possible hit

Fig. 4 – Possible motion-prediction confidences during image-based rendering. The reference image is depicted with a grid and was previously rendered from the view point of the dim yellow camera shown in the lower left corner of (a), (b), and (c). Rays from a new position (shown as a blue camera) are tra versed as lines in the reference frames. Rays pass in front of (pixels upto red pixels along the ray) and behind (blue pixels) sampled geometry. – blue pixel next to red pixel implies a possible intersection of ray and surface.

model renderer as the start point for a true intersection search. The model renderer is responsible for completing the rendering of these pixels.

Aliasing is a problem for all image-based rendering techniques. Adding an object ID at each pixel reduces aliasing in cases where two neighboring reference image pixels with the same object ID are connected by a "smooth" surface. Our definition of "smooth" is informal, meaning simply that interpolated color and depth approximate true color and depth. Surfaces with sharp concavities or protrusions violate the interpolation assumption, however those surface polygons can have different object IDs to indicate this condition. A priori ID labeling may be impractical, (e.g., for volume isosurfaces or very fine mesh data) so a simple strategy locates the problematic regions at run time. If the depths of the two neighboring pixels in the reference image differ by more than a threshold (e.g., a one-voxel distance in the case of volume data), we assume a concavity (or sharp feature) and return a possible hit.



(a) (b)

Fig. 5 - concavity test - A is a reference image camera position, B is a new camera. In case (a) intersections x and y (depths of two neighboring pixels) are within 1 voxel distance (reuse color or reshading). In case (b) x and y are farther than 1 voxel distance so smoot h-ness test fails (cast a ray from given depth)

Another image-based rendering problem arises with moving objects in the scene. Our current implementation only deals with a moving camera although a limited number of moving objects can be handled trivially with separate reference images. In the limit, each object has a separate reference image (or sprite) each rendered independently. Intersections with each reference image are composited like layered images or sprites.

3. COMPRESSION WITH HYBRID IBR

High compression ratios are achieved by reducing the transmission of redundant interframe (temporal) and intraframe (spatial) information. The reference image is the starting point in the IBRAC system, and the motion prediction encodes its approximate temporal evolution. The smoothness assumption in the intersection tests implicitly encodes spatial similarity. The low-confidence possible-hit case indicates where new information is needed. This additional information is computed by the model renderer (R_M) and encoded into a residual image. To achieve high compression ratios, the residual image should contain few pixels or low variations of pixel values. In IBRAC, the number of pixels in the residual image depends on the threshold of smoothness that determines whether surface depth variations warrant a possible-hit intersection test classification as explained in the previous section. This threshold provides a convenient control over the SNR of the compressed images.

IBRAC computes pixel motion and the residual image in image-order. High confidence pixels have their color and depth interpolated from reference images. Pixels with low-confidence motion estimates are computed by the model renderer and

included in the residual image. Since the IBRAC encoder and decoder use the same reference image (i.e., a precomputed frame or the prior frame), no motion estimates need to be transmitted. Also, since the sequence of residual pixels is known to both the encoder and decoder, there is no need to transmit address or index data with the residual image. These unique advantages of IBRAC dramatically raise the compression ratio.

Comparing IBRAC residual data with JPEG (~75% quality) or GIF encodings of the same residual images, shows an order of magnitude or more difference. By not transmitting any motion data, IBRAC compression ratios improve over MPEG-2 by a factor of 2-10 in many cases.

3.1. Compression System

The equations below are a functional expression for the overall compression and decoding process. An index 'i' or 'j' indicates a frame from a sequence. Capital "**R"** identifies a renderer and "**I"** denotes an image. The data flow is illustrated in figure 2. Reference images or *I-frames* ($\mathbf{I_I}$) are generated from the model renderer (Eq. 1). These images contain color, depth, and object-ID at each pixel. *Extracted Images* ($\mathbf{I_E}$) are created by the *Image-Based Renderer* ($\mathbf{R_I}$) from a given I-frame (Eq. 2). The *Residual Image* ($\mathbf{I_R}$) is the set of pixels for which the image-based renderer cannot find good motion estimates for the *Extracted Images* ($\mathbf{I_E}$). *Residual Image* ($\mathbf{I_R}$) pixels are computed by the model renderer ($\mathbf{R_M}$) (Eq. 3). The *Final Image* ($\mathbf{I_F}$) combines an extracted image $\mathbf{I_I}$ and a residual image $\mathbf{I_R}$ (Eq. 4).

$$\mathbf{I_I}[j] = \mathbf{R_M}(\text{Data}_{3D}) \quad \text{(Eq. 1)}$$
$$\mathbf{I_E}[i] = \mathbf{R_I}(\mathbf{I_I}[j]) \quad \text{(Eq. 2)}$$
$$\mathbf{I_R}[i] = \mathbf{R_M}(\text{Data}_{3D}) \cap \mathbf{I_E} \quad \text{(Eq. 3)}$$
$$\mathbf{I_F}[i] = \mathbf{I_E}[i] \cup \mathbf{I_R}[i] \quad \text{(Eq. 4)}$$

The encoder (server) has the entire 3D model, a model renderer ($\mathbf{R_M}$), and an image-based renderer ($\mathbf{R_I}$), and it is capable of creating all the final images ($\mathbf{I_F}$). The decoder (client) needs only the image-based renderer ($\mathbf{R_I}$) and one or more I-frames ($\mathbf{I_I}$), initially received from the server. The decoder generates final images ($\mathbf{I_F}$) by merging the extracted images ($\mathbf{I_E}$) it computes, with the residual images ($\mathbf{I_R}$) received from the server.

The final images at the encoder and decoder are identical ($\mathbf{I_F}$[i] at encoder = $\mathbf{I_F}$[i] at decoder). The compression is lossless in this sense. Thus, the final images could be used as reference images for the next frame. However, the images generated by the model renderer are not identical to those generated by the total system ($\mathbf{I_F}$[i] != $\mathbf{I_I}$[i]), due to the errors introduced by sampled geometry and shading in the image-based renderer.

3.2. Encoding

One or more reference images (with color, depth, and object ID) are rendered initially or produced off-line and stored with the 3D model. The reference and residual images at the encoder are compressed before transmission to the decoder. Residual images have no offset or index values and lossless compression is possible with an entropy coder like [Ziv77]. This compression further increases the compression ratio. As seen in figure 6, the compressed IBRAC residual data is 1/10 and 1/100 the size of a GIF and lossless JPG encoding of the same residual image, respectively. This is indicative of the overall system performance since the transmitted data is mainly residual image data.

3.3. Decoding

The decoding process requires an initial reference image to bootstrap the motion estimation process. The initial reference image contains color, depth, and ID, and it must be transmitted losslessly. This reference image may have lower resolution than the output images to reduce startup bandwidth and latency. Future residual images refine the output images that later serve as reference images themselves. In our implementation, the initial reference image (color, object ID and depth) is compressed with gzip, an implementation of Lempel-Ziv encoding, which showed better compression than lossless JPEG in many cases.

4. Performance Enhancement

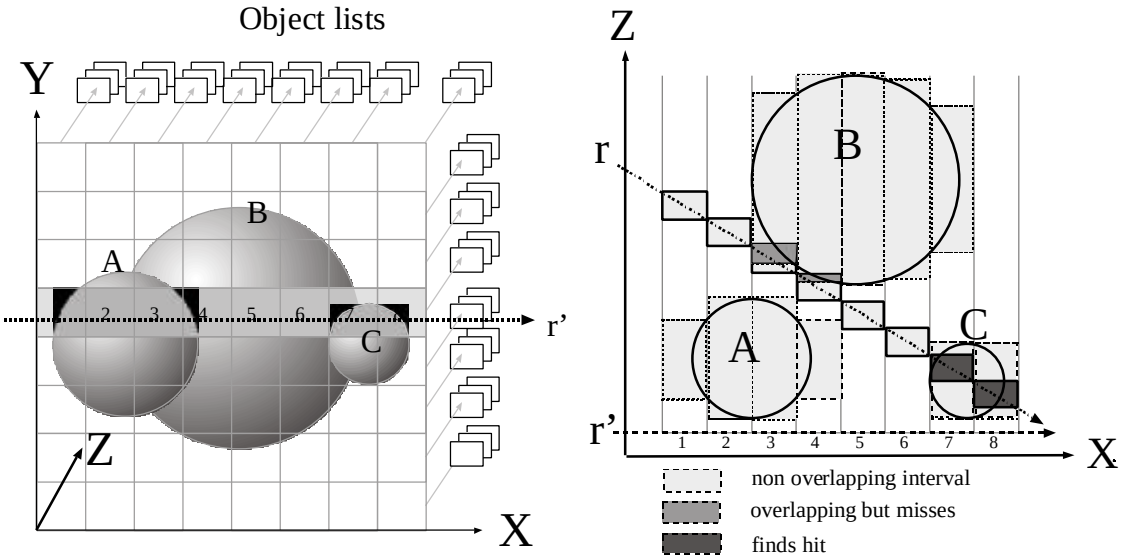
The model renderer and image-based renderer are based on ray casting, which can be substantially improved by employing acceleration techniques. Currently, our implementation employs a method that improves the speed.

4.1 Orthographic layered depth images (OLDI)

An orthographic layered depth image is employed to accelerate the model renderer, especially for polygon models. It is useful to the model renderer when creating the reference image, or when the image-based renderer returns a possible hit. Similarly to layered depth images [Shade98], an orthographic layered depth image is created by orthographically projecting the scene primitives onto it. Each pixel contains a list of all objects that project within its bound (Fig. 5a). The list keeps each

object identifier and its minimum and maximum depth within the pixel's area. This object list is sorted by their minimum depth to speed searches to a given depth during traversal of reference images.

During reference image traversal in the image-based renderer, ray depth is checked against a pixel's minimum and maximum depth. Pixels are checked sequentially along the projected ray. As shown in figure 6b, intersection tests occur only when a ray depth boundary overlaps with an object list depth interval. There may be some cases in which a ray boundary overlaps but does not actually hit the object. This "overlapping but misses" case as shown in Figure 6b. Tighter boundary testing can reduce such cases. However, these tests only suggest which object to test with a ray-object intersection test and additional tests never degrade an image. This OLDI method outperformed a BSP acceleration method by a factor of ten [Demers97] and is easily embedded in our system since it is data structure is closely related to the reference image. In addition, this method may be further improved by spatial subdivision methods employed by the image-based renderer.



(a) orthographic layered depth image (b) object lists of pixels in a single scan line r'

Fig. 6 - r is a view ray that is parallel to the x-axis and r' is projection of ray r onto the orthographic layered depth image.

5. RESULTS

Test data includes polygon data of varied complexity and volume data. We generated animations by moving the camera within the scene. As seen in plate 1 and figure 12, the test sequences exhibit rapid viewpoint changes over several different data types. Figure 8 compares compression ratio of different encodings such as lossless JPEG, lossy JPEG at 75% quality, GIF and IBRAC of the same residual images and compare lossless JPEG, lossy JPEG at 75% quality, and GIF encoding of complete images. Use of the IBRAC encoding yields a compression ratio over ten times higher than the alternatives for the same residual images. It also compares compression of the residual image and complete images, showing how sparse the residual is. Therefore, transmitting a frame by IBRAC is 30 - 200 times better than simply transmitting a frame (as a complete image, not a residual) by lossless JPEG and 10 - 70 times better than lossy JPEG at 75% quality. We also compared the image sequence shown in figure 12 with a lossy MPEG-2 encoding (using a Berkley encoder, version 1.2, at default quality). The animations were encoded using the data of Table 1a on an SGI Onyx2 R10K 195 MHz CPU. The IBRAC method achieved a per frame compression ratio about 2 to more than 10 times better than the MPEG2 encoding (Fig. 10 and Fig. 11). As shown in figure 12, even with substantial camera movements, our residual images are small whereas the MPEG compression ratio is relatively poor in comparison. This characteristic arises because the motion prediction is robust for large motions because of the knowledge of 3D positions. The residual images remain sparse as long as the camera remains trained in the vicinity of a reference image, regardless of zoom, rotation, or translation.

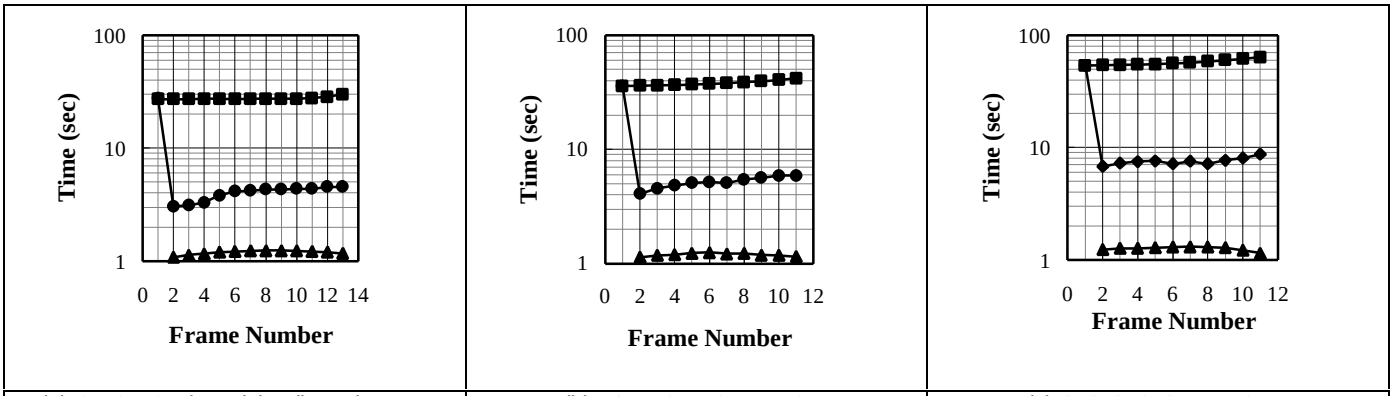
High compression leads to high acceleration since only residual pixels are rendered by the model renderer, and the image-based renderer is much faster and independent of data size. We tested the same volume data in three different sizes by interpolating additional points for a fair comparison. As shown in figure 8, the image-based renderer on the server side accelerates rendering speed by a factor of 3-9 and the client image-based renderer shows execution time independent of data size, accelerating its rendering speed by a factor of 20-30 (Fig. 7).

The image quality at the local renderer and the remote renderer are the same, as explained before. Compared to a stand-alone image-based renderer, our images are better since the hybrid system integrates model rendering to create the residual images that substantially improve image quality. However, IBRAC is a lossy compression method. The interpolated depth and color from reference images is different from what the model renderer would compute. Diffuse surfaces are more forgiving than specular ones. These problems are encountered with other image-based-rendering methods and discussed in [Chen93] [McMillan95]. However, we find these artifacts to be subjectively minor and objectively, the images exhibit a better PSNR (peak signal to noise ratio, calculated as 10 $\log_{10}(255^2/\text{MSE})$) than the MPEG-2 encodings in most frames. Figure 10 and 11 shows the PSNR for each frame of the animation sequences. Images in figure 12, plate 1 and animations (found at http://www-scf.usc.edu/~iyoon/html/ research.html) show little visible degradation. Figure 11 shows better overall PSNR than figure 10 since the model for figure 11 has a less specular surface.

The initial reference image is compressed without loss, so it has infinite SNR. The number of pixels in the residual images depends on the threshold of smoothness that determines whether surface depth variations warrant a possible-hit intersection-test classification as explained in the previous section. This threshold provides a convenient control over the PSNR of the compressed images as a trade off between image quality and compression ratio or acceleration (Fig. 9).

	Data		Result		
	Type	Size	Image size	Bits/pixel	PSNR
(a)	Volume	128x128x84x4	512x512	0.0256	41.78 dB
(b)	Volume	256x256x128x4	256x256	0.0707	31.8 dB
(c)	Polygons	897triangles	512x512	0.016	38.17 dB
(d)	Polygons	33264 triangles	400x400	0.0454	37.04 dB
(e)	Volume	64x64x64x4	256x256	0.077515	35.64 dB

Table 1 – Size of polygon and volume data for result images shown in Plate 1.



(a) 64x64x64 (voxels)x4(bytes) = 1M | (b) 127x127x127x4 = 8M | (c) 253x253x25x4 = 64M

Fig. 7 - Rendering acceleration test with different data size. Rectangular dots represent the time taken without acceleration (using only model renderer). Circle dots represent time taken in the server image-based renderer, and triangular dots represent time taken in the client.

Figure 10 and 11 illustrate IBRAC acceleration of rendering speed, compression ratio, and SNR for the Chevy model (Table 1d) and volume data (Table 1e). The IBRAC method shows a speedup of three or more in encoding (server side) and thirty or more in decoding (client). Decoding is faster since the residual is received rather than computed. As expected, the rendering time for the remote client side (decoding) is dependent on the image size, rather than the data set complexity.

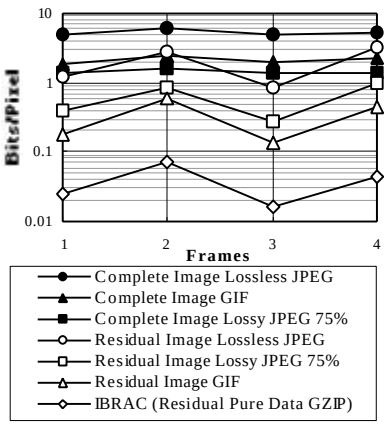


Fig. 8 - Compression ratio comparison of a residual image (lossless JPG/ lossy JPG at 75% quality, GIF, and IBRAC) and complete image (lossless JPG/ lossy JPG at 75% quality, GIF).

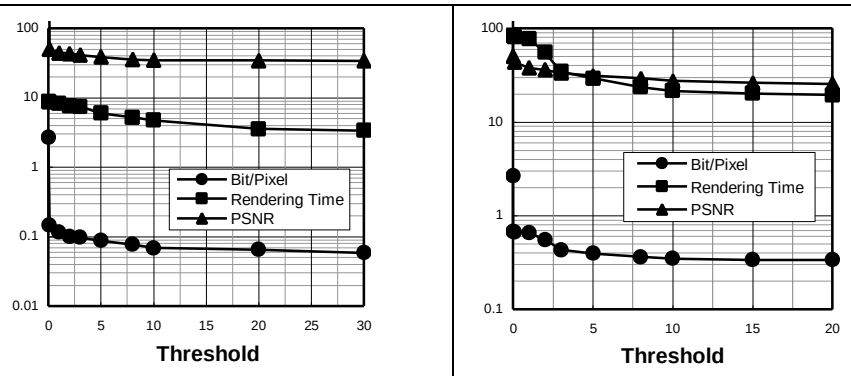
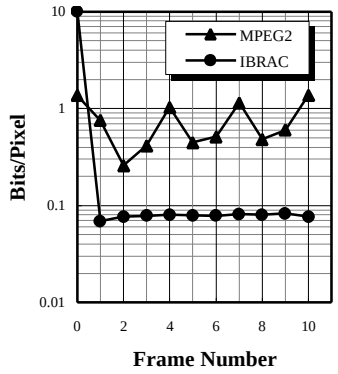
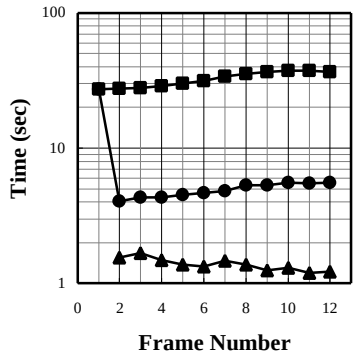


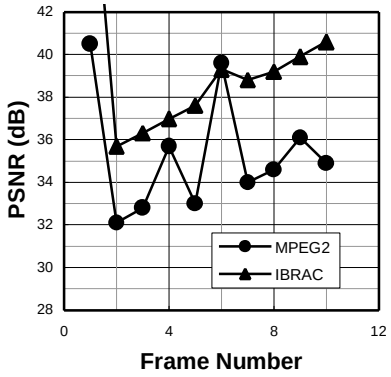
Fig. 9 - Threshold control of PSNR vs. compression (Bits/Pixel) & acceleration (se c-onds). The left chart tests the volume data described in Table1 (e). The right chart is produced with the Chevy polygonal data set.



(a) Compression ratio comparison with MPEG2

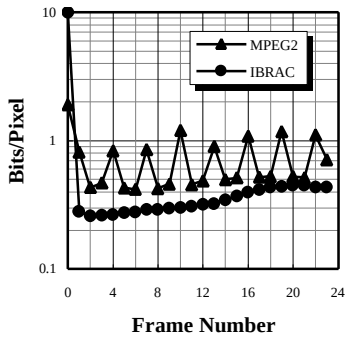


(b) Rendering acceleration

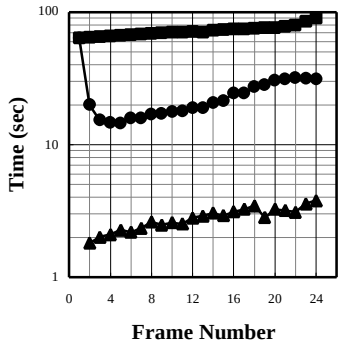


(c) SNR comparison with MPEG2

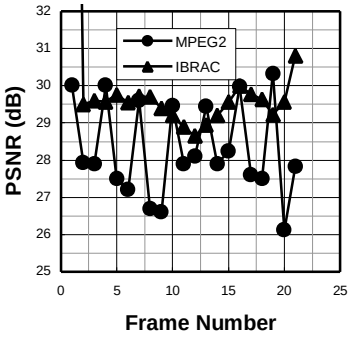
Fig. 10 - Volume data animation tests



(a) Compression ratio comparison with MPEG2



(b) Rendering acceleration



(c) SNR comparison with MPEG2

Fig 11 - Chevy data animation tests

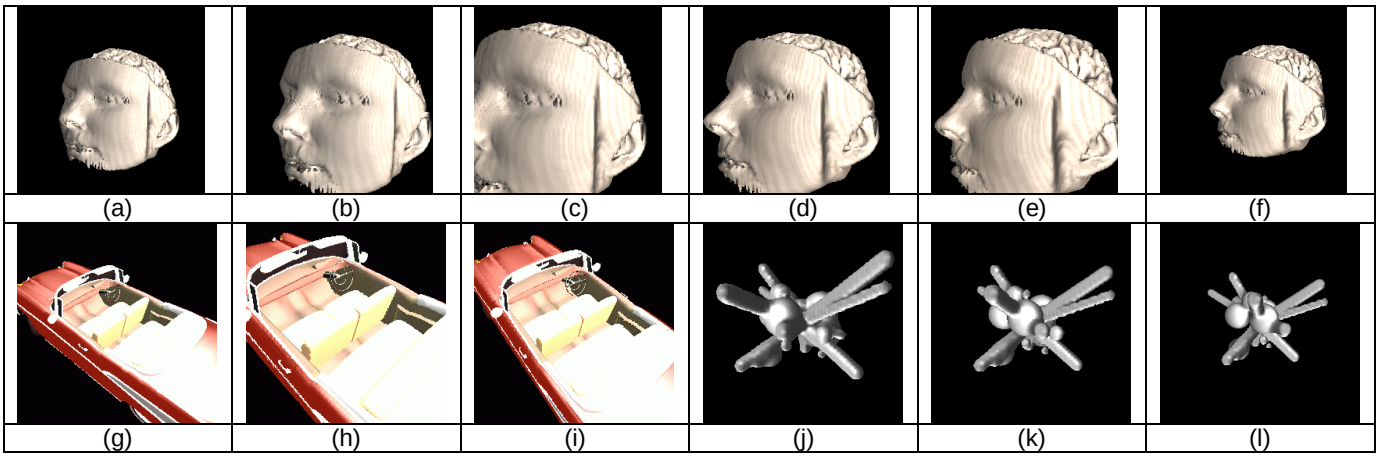


Fig. 12 - Images from the test animation sequences. Note the use of scaling and rotation. Mpeg animations made from the frames created by the IBRAC method are available at http://www-scf.usc.edu/~iyoon/html/research.html .

6. DISCUSSION AND FUTURE RESEARCH

This paper presents an extension of image-based rendering methods to compression. Using a reference image, new views of the scene are estimated and residual images are compressed into less than 0.1 bits/pixel, in many cases. This performance comes from robust motion estimation and confidence assessment. The IBRAC method is effective for generating and compressing synthetic image sequences in remote rendering and internet applications especially in cases of large 3D data sets.

The current implementation supports only a single reference image. Future work will explore block-based multiple reference images, varied resolution reference images, and additional controls over quality and bandwidth tradeoffs.

7. ACKNOWLEDGEMENTS

This research has been funded by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center with additional support from the Annenberg Center for Communication at the University of Southern California and the California Trade and Commerce Agency. We thank Antonio Ortega, (EE Dept.) for reviewing the text and we thank Doug Fidaleo and Clint Chua for editing assistance. Special thanks go to Joe Demers and Taeyong Kim for many of the initial ideas and guidance in the implementation.

8. REFERENCES

[Blinn76] Blinn, J.F., Newell, M.E., “Texture and Reflection in Computer Generated Images,” CACM, Vol 19, No 10, Oct. 1976 pp 542 – 547

[Blinn78] Blinn, J.F., “Simulation of Wrinkled Surfaces”, (Proc. SIGGRAPH 78, pp 286-292)

[Chen93] Chen, E., and Williams, L., “View Interpolation for Image Synthesis”, Computer Graphics (Proc. SIGGRAPH 93, pp 279-288)

[Cohen-Or97] Cohen-Or, D., "Model-Based View Extrapolation for Interactive VR Web-Systems," Computer Graphics International, 1997.

[Cook84] Cook, R.L., “Shade Trees”, (Proc. SIGGRAPH 84, pp 223-231)

[Debevec96] Debevec, P., Taylor, C.J. and Malik, J., “Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach,” (Proc. SIGGRAPH 96, pp 11-20)

[Demers96] Demers, J., Yoon, I., Kim, T.Y., and Neumann, U., " Accelerating Ray Tracing by Exploiting Frame-to-Frame Coherence," Technical Report of USC, available at http://www-scf.usc.edu/~iyoon/html/research.html.

[Djurcilov98] Djurcilov, S., Pang, A., "Visualization Products On-Demand Through the Web", VRML 98 Third Symposium on the VRML, pp 7 - 24

[Earnshaw97] The Internet in 3D Informarion, Images and Interaction / edited by Earnshaw, R. and Vince, J., Academic Press, 1997

[Greene86] Greene, N., “Environment Mapping and Other Applications of World Projections,” IEEE CG&A, Vol 6 Nov. 1986 pp.21-29

[Guenter93] Guenter, B.K., Yun, H.C. and Mersereau, R.M., “Motion compensation compression of computer animation frames”, Computer Graphics (Proc. SIGGRAPH 93, pp 297-304)

[Hidaka90] Hidaka, T., Ozawa, K., “Subjective assessment of redundancy-reduced moving images for interactive applications: Test methodology and report,” Signal Processing: Image Commun. 2, 2 (Aug. 1990)
[JPEG] JPEG digital compression and coding of continuous-tone still images. Draft ISO 10918. 1991
[Kajiya84] Kajiya, J.T. and von Herzen, B.P., “Ray tracing Volume Densities”, Computer Graphics, (Proc. SIGGRAPH 84 pp 165-174)
[Laveau94] Laveau, S. and Faugeras, O.,“3-D Scene Representation as a Collection of Images”, Technical Report RR-2205, http:www.inria.france
[Levoy95] M. Levoy, “Polygon-Assisted JPEG and MPEG Compression of Synthetic Images”, Computer Graphics, (Proc. SIGGRAPH 95 pp 21-28)
[Levoy96] Levoy, M., Hanrahan, P., “Light Field Rendering”, Computer Graphics, (Proc. SIGGRAPH 96 pp 31-41)
[Lengyel97] Lengyel, J. and Snyder, J., “Rendering with Coherent Layers”, (Proc. SIGGRAPH 97 pp 233-242)
[Li98] J. Li, "Progressive Compression of 3D graphics," Ph.D Dissertaion, University of Southern California, May 1998
[Liou91] Liou, M.L., “Overview of the px64 kbps video coding standard,” Commun. ACM 34, 4 (Apr. 1991)
[Mann97] Mann, Y. and Cohen-Or, D., "Selective Pixel Transmission for Navigating in Remote Virtual Environments", Eurographics '97, Volume 16, Number 3.
[Mark97] Mark, W.R., McMillan, L. and Bishop, G., "Post-Rendering 3D Warping", Symposium on Interactive 3D Graphics, 1997, pp 7-16.
[McMillan95] McMillan, L. and Bishop, G. ,“Plenoptic Modeling: An Image-Based Rendering System”, Computer Graphics, (Proc. SIGGRAPH 95, pp 39-46)
[McMillan97] McMillan, L., “An Image-based Approach to Three-Dimensional Computer Graphics”, Ph.D Dissertation, University of North Carolina (UNC-CS Tech. Report TR97-013), April 1997.
[MPEG] MPEG proposal package description. Document ISO/WG8/MPEG/89-128 (July 1989)
[Rabbani91] Rabbani, M. and Jones, P.W., 1991. “Digital Image Compression Techniques”, SPIE Optical Engineering Press, Bellingham, WA.
[Regan94] Regan, M. and Pose, R., “Priority Rendering with a Virtual Reality Address Recalculation Pipeline”, (Proc. SIGGRAPH 94, pp 155-162)
[Shade96] Shade, J. Lischinski, D., Salesin, D., DeRose, T. and Snyder J., “Hierarchical image caching for Accelerated Walkthroughs of Complex Environments”, (Proc. SIGGRAPH 96, pp 75-82)
[Torborg96] Torborg, J., Kajiya, J.T., “Talisman: Commodity Realtime 3D Graphics for the PC,” (Proc. SIGGRAPH 96, pp 353-364)
[VHM] Visible Human Project web page http://www.nlm.nih.gov/research/visible/visible_human.
[VRML] ISO/IEC 14772-1, The Virtual Reality Modeling Language, http://www.vrml.org/VRML97/DIS/, 4 April 1997
[Yagel93] R. Yagel and Z. Shi, "Accelerating Volume Animation by Space-leaping," Proceedings of Visualization '93, San Jose, California, Oct. 1993, pp 62-69
[Yoon97] Yoon, I., Demers, J., Kim, T.Y. and Neumann, U., “Accelerating Volume Visualization by Exploiting Temporal Coherence”, (Proc. IEEE Visualization 97 LBHT, pp 21-24)
[Yun97] Yun, H.C., Guenter, B.K., and Mersereau, R.M. “Lossless Compression of Computer-generated Animation Frames”, ACM Transactions on Graphics Vol 16, No. 4, October 1997, pp 359-396
[Ziv77] Ziv, J., Lempel, A., “A universal algorithm for sequential data compression,” IEEE Transactions on Informaion Theory, IT-23:337-343, 1977

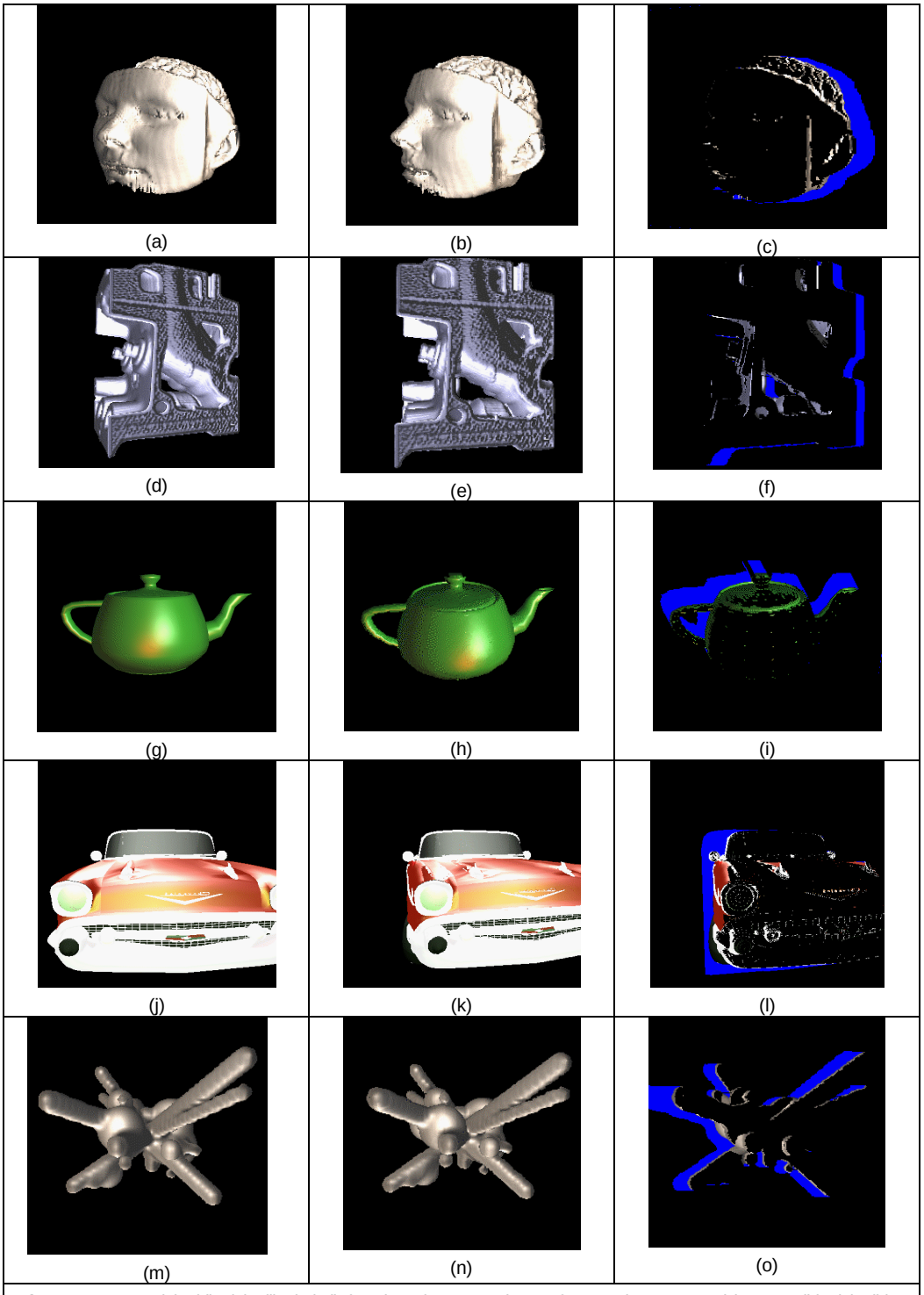


Plate1 - Images (a), (d), (g), (j), (m) (left column) are used as reference images, and images (b), (e), (h), (k), (n) (center column) are images generated from $\mathbf{R}_I$ (image-based renderer) and (c), (f), (i), (l) , (o) (right column) are residual images. The blue regions show pixels with low confidence motion estimates from $\mathbf{R}_I$, that did not intersect objects or surfaces. The colored regions show low confidence pixels that were re n-dered by the model renderer $\mathbf{R}_M$.